SPIM & MIPS

Department of Information and

Management

Dan

Ming-Shiuan Chen

TAs





TA

• Name: 陳明軒 (包子)

Ming-Shiuan Chen

- Gender : Male
- Email :

intere2960@cmlab.csie.ntu.edu.tw





Name:廖以圻(Chi-Chi) Chi-Chi Lao
Gender:Male
Email:

chichi@cmlab.csie.ntu.edu.tw



Outline

Assembly Language
SPIM
MIPS
Homework 2



Assembly Language

Computer Organization and Structure 2015

Page 6

Assembly Language

binary instructions

Translates assembly language into

Assembly language
 Symbolic representation of a computer's binary encoding
 Assembler

Machine code

Computer's binary encoding

Assembly Language



FIGURE B.1.6 Assembly language either is written by a programmer or is the output of a compiler.

Computer Organization and Structure 2015

Why Assembly

• A low level language

the code and syntax is much closer to the computer's processor

- Direct hardware manipulation device drivers, low-level embedded systems, and real-time systems
- Speed optimization performance and efficiency



 To write in assembly is to understand exactly how the processor and memory work together to "make things happen".



Sometimes to debug a higher-level language, you have to review the resulting assembly language.



SPIM

Computer Organization and Structure 2015

Page 11

(11/7)

What is SPIM

• MIPS32 Simulator

reads and executes assembly language program written for MIPS 32-bit architecture

- SPIM does not execute binary programs provides a simple debugger and minimal set of operating system services
- SPIM implements both a terminal

QtSPIM Installation

SPIM: A MIPS32 Simulator

James Larus spim@larusstone.org

Contents

- Older Versions of SPIM
- Further Information
- <u>Changes to SPIM</u>
- <u>Copyright</u>

Spim is a self-contained simulator that runs MIPS32 programs. It reads and executes assembly language programs written for this processor. Spim also provides a simple debugger and minimal set of operating system services. Spim does <u>not</u> execute binary (compiled) programs.

Spim implements almost the entire MIPS32 assembler-extended instruction set. (It omits most floating point comparisons and rounding modes and the memory system page tables.) The MIPS architecture has several variants that differ in various ways (e.g., the MIPS64 architecture supports 64-bit integers and addresses), which means that *Spim* will not run programs for all MIPS processors.

Spim comes with complete source code and documentation.

Spim implements both a terminal and windows interfaces. On Microsoft Windows, Linux, and Mac OS X, the *spim* program offers a simple terminal interface and the *QtSpim* program provides the windowing interface. The <u>older</u> programs *xspim* and *PCSpim* provide window interfaces for these systems as well.

Download SPIM

What's New?

QtSpim is a new user interface for *Spim* built on the <u>Qt UI framework</u>. Qt is cross-platform, so the same user iterface and same code will run on Windows, Linux, and Mac OS X (yeah!). Moreover, the interface is clean and up-to-date (unlike the archaic X windows interface).

11

Page 13

Spim has moved to <u>SourceForge</u>! The source code for all version of Spim are in an SVN repository and compiled version are available for download. There is also a bug tracker and discussion forum. Spim is an open source project, so please join in and contribute.

QtSPIM Installation

spim mips simulator

Brought to you by: jameslarus

Reviews Support Wiki Code Tickets • Summary Files

Looking for the latest version? Download QtSpim_9.1.12_Windows.exe (31.5 MB)

Home

Name 🕈	Modified \$	Size +	Downloads / Week +	
qtspim_9.1.16_linux32.deb	2015-08-26	29.7 MB	71 🔔	0
QtSpim_9.1.16_mac.mpkg.zip	2015-08-26	28.7 MB	248 💶	0
qtspim_9.1.16_linux64.deb	2015-08-26	28.1 MB	137 🔔	0
QtSpim_9.1.16_Windows.exe	2015-08-26	34.3 MB	1,014 🔟	0
QtSpim_9.1.15_mac.mpkg.zip	2015-04-26	28.7 MB	128 🔔	0
QtSpim_9.1.13.mac.mpkg.zip	2014-02-05	33.1 MB	2	0
QtSpim_9.1.12_Windows.exe	2013-12-17	31.5 MB	547 🛌	0
qtspim_9.1.12_linux32.deb	2013-12-14	1.1 MB	10	0
qtspim_9.1.12_linux64.deb	2013-12-14	1.1 MB	120 🔔	0
qtspim_9.1.9_linux64.deb	2013-01-23	1.1 MB	3	0

2

11170

Page 14

Computer Organization and St

QtSPIM Screenshot

R QtSpim – 🗖 🗙		
File Simulator Registers Text Segment Data Segment Window Help	100	
	100	
FP Regs Int Regs [16] Data Text	- 22	
Int Regs [16] B × Text	• >	
PC = 400020 ^ User Text Segment [00400000][00440000]	^	
$EPC = 0 \qquad [00400000] 8fa40000 lw $4, 0($29) ; 183: lv $a0 0($sp) # argc$		
Cause = 0 [00400004] 27a50004 addiu \$5, \$29, 4 ; 184: addiu \$a1		
BadVAddr = 0 [00400008] 24a60004 addiu \$6, \$5, 4 ; 185: addiu \$a2		
Status = 3000ff10 [0040000c] 00041080 sll \$2, \$4, 2 ; 186: sll \$v0 \$		
[00400010] 00c23021 addu \$6, \$6, \$2 ; 187: addu \$a2		
HI = 0 [00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main		
10 = 0 [00400018] 00000000 nop ; 189: nop Hello World!! This is Ming-Shiuan		
[0040001c] 3402000a ori \$2, \$0, 10 ; 191: li \$v0 10		
R0 [r0] = 0 [00400020] 000000c syscall ; 192: syscall #		
$\begin{bmatrix} a c \\ b c \end{bmatrix} = 0 $ $\begin{bmatrix} 00400024 \end{bmatrix} 34020004 \text{ ori } 2, 0, 4 $ $i 40: 1i 5v0, 4$		
R3 [v1] = 0 [00400028] 3c041001 lui \$4, 4097 [msg] ; 41: la \$a0, ms		
R4 [a] = 10010000 [0040002c] 000000c syscall ; 42: syscall #		
R5 [a1] = 7ffff244 [00400030] 8f898000 lw \$9, -32768(\$28) ; 43: lv \$t1, fo		
R6 [a2] = 7ffff24c [00400034] 03e00008 jr \$31 ; 45: jr \$ra # r		
R7 [a3] = 0		
R8 [t0] = 0 Kernel Text Segme		
R9 [t1] = 0 [80000180] 0001d821 addu \$27, \$0, \$1 ; 90: move \$k1 \$		
R10 [t2] = 0 [80000184] 3c019000 lui \$1, -28672 ; 92: sv \$v0 s1		
R11 [t3] = 0 [80000188] ac220200 sw \$2, 512(\$1)		
R12 [t4] = 0 [8000018c] 3c019000 lui \$1, -28672 ; 93: sv \$a0 s2		
R13 [t5] = 0 [80000190] ac240204 sw \$4, 516(\$1)		
R14 [t6] = 0 [80000194] 401a6800 mfc0 \$26, \$13 ; 95: mfc0 \$k0 \$		
R15 [t7] = 0 [80000198] 001a2082 srl \$4, \$26, 2 ; 96: srl \$a0 \$k		
R16 [SU] = 0 [8000019c] 3084001f andi \$4, \$4, 31 ; 97: andi \$a0 \$		
R17 [S1] = 0 [800001a0] 34020004 ori \$2, \$0, 4 ; 101: li \$v0 4		
$\begin{bmatrix} 12 & -0 \\ 10 & -0 \end{bmatrix} = \begin{bmatrix} 800001a4 \end{bmatrix} 3c049000 \texttt{lui $4, -28672 [_m1_]}; 102: la $a0 __bccccccccccccccccccccccccccccccccccc$		
[800001a8] 000000c syscall ; 103: syscall		
R21 [s5] = 0 [800001ac] 34020001 ori \$2, \$0, 1 ; 105: li \$v0 1		
Image: Rest [sol] = 0 [800001b0] 001a2082 srl \$4, \$26, 2 ; 106: srl \$a0 \$		
All Dights Deserved		
SPIM is distributed under a BSD license.		
See the file README for a full copyright notice.		
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 🛄 📖 📖 💷		
R27 [k1] = 0 v [00000104] 30013000 101 \$1, -20072 ; 112: 1v \$a0excp(\$a0)	~	

Computer Organization and Structure 201.

Page 15

References of SPIM

- Official website of SPIM: <u>http://spimsimulator.sourceforge.net/</u>
- Assemblers, Linkers, and the SPIM Simulator: http://pages.cs.wisc.edu/~larus/HP_AppA.pdf
- MIPS Instruction Reference: http://www.mrc.uidaho.edu/mrc/people/jff/d igital/MIPSir.html

MIPS

Computer Organization and Structure 2015

Page 17

(11/7)

MIPS memory layout

- MIPS 32-bit CPU (all registers are 32 bits wide) accessible memory range: 0x00000000–0xFFFFFFFF
- Memory holds both instructions (text) and data If a program is loaded into SPIM, its .text segment is automatically placed at 0x00400000,its .data segment at 0x1000000

MIPS Assembly

Operation Code (Opcode)

Arthmetic Instructions	 add, sub, addi, addu, addiu, subu
Data Transfer Instructions	 Iw, sw, Ibu, sb, Iui, ori
Logic Instructions	 beq, bne, slt, slti, sltu
Branch and Jump- Related Instructions	• j, jr, jal

MIPS Assembly

MIPS Registers and Usage Convention)

\$zero	constant 0
\$v0, \$v1	expression of a function
\$a0 ~ \$a3	argument 1~4
\$t0 ~ \$t9	temporary registers
\$s0 ~ \$s7	save registers
\$sp	stack pointer
\$fp	frame pointer
\$ra	return address
•••	•••

MIPS Assembly

Some data types in MIPS

.word, .half	32/16 bit integer
.byte	8 bit integer
.ascii, .asciiz	string
.double, .float	floating point

11171

Page 21

Assembler Syntax

• Comment : (#)

Everything from the sharp sign to the end of the line is ignored

Identifier : (A sequence of alphanumeric characters

, _ , and .)

Identifier are a sequence of alphanumeric characters, underscores (_), and dots (.) that do not begin with a number

• Instruction Opcode

Instruction opcodes are reserved words that are not valid identifiers

• Label

Labels are declared by putting them at the beginning of a line followed by a colon.

MIPS—Hello World

C VS MIPS

int main()

printf("Hello World¥"); return 0;

.data Mystr: .asciiz "Hello World¥n" .text main: li \$v0, 4 la \$a0, Mystr syscall li \$v0, 10 syscall

MIPS – Hello World

MIPS Architecture

Put Static Data Here

Put Your Code Here

.data

Mystr: .asciiz "Hello World¥n" Yourint: .word 75 Hisarray: .word 100, 100, 100 .word 20, 40, 60 .word 1, 2, 3

.text

••••

Page 24



Page 25

MIPS System Calls

- SPIM provides a small set of operating-system-like services through the system call instruction.
- A program loads the system call code into register \$v0 and arguments into registers \$a0-\$a3
 (or \$f12 for floating-point values).
- System calls that return values put their results in register \$v0 (or \$f0 for floating-point results).

MIPS System Calls

Service	System call code	Arguments	Result
print_int	1	\$a0 = integer	
print_float	2	\$f12 = float	
print_double	3	\$f12 = double	
print_string	4	\$a0 = string	
read_int	5		integer (in \$v0)
read_float	6		float (in \$f0)
read_double	7		double (in \$f0)
read_string	8	\$a0 = buffer, \$a1 = length	
sbrk	9	\$a0 = amount	address (in \$v0)
exit	10		
print_char	11	\$a0 = char	
read_char	12		char (in \$a0)
open	13	<pre>\$a0 = filename (string), \$a1 = flags, \$a2 = mode</pre>	file descriptor (in \$a0)
read	14	\$a0 = file descriptor, \$a1 = buffer, \$a2 = length	num chars read (in \$a0)
write	15	<pre>\$a0 = file descriptor, \$a1 = buffer, \$a2 = length</pre>	num chars written (in \$a0)
close	16	\$a0 = file descriptor	
exit2	17	\$a0 = result	

Computer Organization ar

FIGURE / \$v0

Page 27

MIPS System Calls - Example

		Service	System can cou
Output: "the ans	swer = 5"	print_int	1
		print_float	2
.data		print_double	3
		print_string	4
str:		read_int	5
asciiz "the	answer = "	read_float	6
		read_double	7
.text		read_string	8
	# system call code for print_str	sbrk	9
	# system can code for print_str	exit	10
la ŞaO, str	# address of string to print	print_char	11
syscall	therint the string	read_char	12
systan	# print the string	open	13
li \$v0, 1	# system call code for print_int	nood	14
	# integrate print	reau	14
II ŞaU, S	# Integer to print	write	15
syscall	# print it		
, にたいの 10		close	16
II \$VU, 10	#system call code for exit	exit2	17
syscall		FIGURE	A 9 1 System
	Output: "the and .data str: .asciiz "the .text li \$v0,4 la \$a0, str syscall li \$v0,1 li \$a0,5 syscall li \$v0,10 syscall	Output: "the answer = 5" .data str: .asciiz "the answer = " .text li \$v0,4 # system call code for print_str la \$a0, str # address of string to print syscall # print the string li \$v0,1 # system call code for print_int li \$a0,5 # integer to print syscall # print it li \$v0,10 #system call code for exit syscall	Output: "the answer = 5" <pre>print_int</pre> .data <pre>print_float</pre> str: .asciiz "the answer = " .asciiz "the answer = " <pre>read_int</pre> .text <pre>read_string</pre> li \$v0_4 # system call code for print_str la \$a0, str # address of string to print syscall # print the string li \$v0_1 # system call code for print_int li \$v0_1 # system call code for print_int read <pre>verite</pre> li \$v0_1 # system call code for print_int read <pre>verite</pre> li \$v0_1 # system call code for print_int read <pre>verite</pre> li \$v0_10 # system call code for exit syscall # print it li \$v0_10 # system call code for exit syscall # print it

Computer Organization and Structure 2015

services.

Execute Program in

- 1. Write your own assembly program, and save it as .sfile
- 2. Simulator Reinitialize Simulator
- 3. Open your .s file 📷
- 4. Simulator Clear Registers 💈
- 5. Simulator Run / Continue

Homework 2

Computer Organization and Structure 2015

Page 30

Homework 2

- This is an individual assignment
- Plagiarism will be heavily punished
- Write the following three programs in MIPS assembly language. (Must run correctly on SPIM)

Find out all prime numbers

Tower of Hanoi

Greatest Common Divisor

• One bonus program : Binary Search

Documentation (20%)

- Detailed documentation for each program is required
- The following parts must be included:
 - Your name, student ID, and email address
 - Explanation of the design or the flow of each program
 - What you've learnt from writing the programs
- Problems or difficulties you've encountered during writing the programs are nice to be included in the document

Problem 1: Find out all prime numbers

- Input:
 - a positive integer n>1
- Output:
 - All prime numbers which is smaller than n
- Requirements:
 - 1. Print the correct answer.
 - 2. Can do many calculations iteratively
 - 3. The file name is FindPrime.s

	💺 Console	
	input:	
I.	б	
l	output: 2.	
	3	
L	5	
l		
l		-

Problem 2: Tower of Hanoi

- A hanoi tower with 3 rods A,B,C and n disks. Move all the disk from A to C
- Input:
 - a positive integers n, $1 < n \le 5$
- Output:
 - Print all the steps
- Requirements:
 Print the correct step.
 The file name is Hanoi.s

Computer Organization and Structure 2015

🐫 Console

input: 3 output: move a to c move a to b move c to b move a to c move b to a move b to c move a to c

Page 34

Problem 3: GCD

• Input :

two positive integers x, y > 1

• Output :

the greatest common divisor of x and y. Output gcd(x, y)

• Requirements:

Print the correct answer
 Can do many calculations iteratively
 The file name is GCD.s

Optional: Binary Search

• Input:

- 1. n positive integers, where n < 8
- 2. value to search
- Output:
 1.If value can be find, print "Find it.".
 2. If value not in the sequence, print "The value is not in this sequence.".
- Requirements:

Print the correct answer. Computer Theonalmericsubi marser is Bi marse



Submisssion

- Deadline: 11:59 PM, Monday, Oct. 19, 2015
- You must submit at least the following files:
 - FindPrime.s
 - Hanoi.s
 - GCD.s
 - Binary_Search.s (Optional)
 - (Your student id)_hw2_document.pdf
- The attach filename should be like b03xxxxxx.zip
- Email your **zipped** file to TA:

intere2960@cmlab.csie.ntu.edu.tw

Grading Guidelines

Description	For Each Problem
Program runs without error messages	10%
Program executes correctly	60%
Documentation and description	20%
Implementation Detail	10%

Computer Organization and Structure 2015

Deadline

• Late submission policy

10% off from your total score each

day

Contact Information

• TA Hours @ 管院 一館五樓 503-C Chi-Chi Liao(廖以圻) Fir. 8:00~10:00 Ming-Shiuan Chen(陳明軒) Tus. 13:00~15:00 • Contact us if you have any problem Chi-Chi: chichi@cmlab.csie.ntu.edu.tw 包子: intere2960@cmlab.csie.ntu.edu.tw



Thank You For Your Attention

Computer Organization and Structure 2015

Page 41